phaseField User Manual

Updated 16th October, 2014

1 Getting Started

If the repo has been cloned successfully, you have a directory, `phaseField`, with the following sub-directories:

```
applications
    allenCahn
    cahnHilliard
    (other application folders)
include
    dealIIheaders.h
src
    AC.h
    CH.h
    (header files corresponding to other applications)
```

Header files in `src` contain most of the "guts" of the deal.ii implementation of the models, and `include` contains the required deal.ii headers. As indicated above, `applications` has sub-directories for each application, and the app is meant to be made, run, and modified in its directory. The application directory initially contains three files:

```
CMakeLists.txt
main.cc
parameters.h
```

Model parameters can be modified in `parameters.h`, and initial conditions are specified in `main.cc`. This document will discuss how to manipulate these files to use the PRISMS applications.

2 Parameters

This section describes the model parameters that the user can access through the `parameters.h` file in each application folder. These can be divided between model-specific parameters and those generic to all models, and there are three categories of

model-specific parameters: constant coefficients and tensors, bulk free energy functions, and residuals. In the context of their model, constant coefficients are self-explanatory. Bulk free energy functions and their derivatives are incorporated as functions of the phase field variables, usually instantiated in the code as `n` or `c`. All of the examples and defaults use polynomial free energies, and their derivatives are found symbolically. Non-polynomial functions could be implemented, but are not supported at this time. Residuals are used directly in evolution. Changing them is not recommended unless the user is comfortable with deal.ii and the PRISMS framework.

2.1 Generic Parameters

- `problemDIM`
  Dimension of the problem (e.g. 1D, 2D, 3D)

- `spanX`
  Length of system in x-direction

- `spanY`
  Length of system in y-direction. Not used if `problemDIM` < 2

- `spanZ`
  Length of system in z-direction. Not used if `problemDIM` < 3

- `refineFactor`
  Defines the refinement of the mesh. There are $2^{\text{refineFactor}}$ elements in each direction in this implementation, and $\left(2^{\text{refineFactor}}\right)^{\text{problemDIM}}$ elements in total.

- `finiteElementDegree`
  The order of interpolation of the finite element space. In this case, the order of the Lagrange elements to be used.

- `dt`
  The simulation timestep.

- `numIncrements`
  The number of simulation iterations. Final time is then `dt`·`numIncrements`.

- `writeOutput`
  Whether we are writing any output. Takes a boolean argument, e.g. `true`.

- `skipOutputSteps`
  Output will be written every `skipOutputSteps` iterations. If `writeOutput` `true`, the initial conditions will always be written.

2.2 `allenCahn` Parameters

Given the evolution equation:

$$\frac{\partial \eta}{\partial t} = -M_\eta \left(f'(\eta) - \epsilon_\eta^2 \nabla^2 \eta \right) \tag{1}$$

- `Mn`
  Allen-Cahn mobility parameter M_η in Eq. 1.

- `Kn`
  Square of the Allen-Cahn gradient energy parameter, ϵ_η^2 in Eq. 1. (tensor form ????)

- `fnV`
  First derivative of the Allen-Cahn bulk free energy $f(\eta)$, where η is represented in code by `n`. The default for $f(\eta)$ in the code is the double well potential:

$$f(\eta) = \eta^2(1-\eta)^2 \tag{2}$$
$$f'(\eta) = 4\eta(\eta-1)(\eta-1/2)$$

- Residuals
  In the context of Eq. 1, we have

$$\texttt{rnV} = \eta - \Delta t \cdot M_\eta f'(\eta)$$
$$\texttt{rnxV} = \nabla \eta$$

  where Δt is the simulation timestep.

2.3 `cahnHilliard` Parameters

Parameters for Cahn-Hilliard are defined in much the same way as for Allen-Cahn. Given the evolution equation:

$$\frac{\partial c}{\partial t} = \nabla M_c \cdot \nabla \left(f'(c) - \epsilon_c^2 \nabla^2 c \right) = \nabla M_c \cdot \nabla \mu \tag{3}$$

- `Mc`
  Cahn-Hilliard mobility parameter M_c in Eq. 1.

- `Kc`
  Square of the Cahn-Hilliard gradient energy parameter, ϵ_c^2 in Eq. 1. (tensor form ????)

- `fcV`
  First derivative of the Allen-Cahn bulk free energy $f(c)$, where c is represented in code by `n`. The default for $f(c)$ is the same double well potential (Eq. 2) as in Allen-Cahn, now in c.

$$f(c) = c^2(1-c)^2$$
$$f'(c) = 4c(c-1)(c-1/2)$$

- Residuals
  In the context of Eq. 3, we have

$$\texttt{rmuV} = f'(c)$$

$$\texttt{rmuxV} = \nabla c$$

$$\texttt{rcV} = c$$

$$\texttt{rcxV} = \nabla \mu$$

2.4 `mechanics` Parameters

These parameters focus on building the stiffness tensor C_{ijkl} (`CijklV`), starting from a Young's Modulus (`Ev`) and Poisson's ratio (`nuV`).

$$\mu = \frac{E}{2(1+\nu)}$$

$$\lambda = \frac{\nu E}{(1+\nu)(1-2\nu)}$$

$$C_{ijkl} = \mu(\delta_{ik}\delta_{jl} + \delta_{il}\delta_{jk}) + \lambda\delta_{ij}\delta_{kl}$$

2.5 `coupledCahnHilliardAllenCahn` Parameters

This application implements the following set of equations for a conserved order parameter, c, and one or more non-conserved (structural) order parameters η:

$$\frac{\partial \eta}{\partial t} = -M_\eta \left(\frac{\partial f(\eta, c)}{\partial \eta} - K_{ij} \nabla_i \nabla_j \eta \right) \tag{4}$$

$$\frac{\partial c}{\partial t} = \nabla \cdot M_c \nabla \left(\frac{\partial f(\eta, c)}{\partial c} \right) \tag{5}$$

Since there is no gradient energy term in c, this might also be called a coupled Allen-Cahn/diffusion system. From a finite element perspective, Eq. 5 is more tractable as

$$\frac{\partial c}{\partial t} = \nabla \cdot M_c \left(\frac{\partial^2 f(\eta, c)}{\partial c^2} \nabla c + \frac{\partial^2 f(\eta, c)}{\partial \eta \partial c} \nabla \eta \right) \tag{6}$$

- `numStructuralOrderParameters`
  Multiple structural/non-conserved order parameters can be used in this model (implemented?????)

- `Mc`
  M_c, mobility of the conserved order parameter c.

- `MnVals`
  M_η, mobility of the non-conserved order parameter η. This is an array with `numStructuralOrderParameters` elements, each corresponding to an order parameter.

- `KnVals`
  K_η, gradient energy coefficients K of the non-conserved order parameter η. Again, an array with `numStructuralOrderParameters` elements, each corresponding to an order parameter.

The free energy $f(\eta, c)$ is composed of free energies $f_\alpha(c)$ and $f_\beta(c)$ for the phases corresponding to $\eta = 1$ and $\eta = 0$, and an interpolating function $h(\eta)$. The default case uses free energies for a Mg-Gd system:

$$f_\alpha(c) = -1.6704 - 4.776c + 5.1622c^2 - 2.7375c^3 + 1.3687c^4 \tag{7}$$

$$f_\beta(c) = -1.5924 - 5.9746c + 5.0c^2 \tag{8}$$

These free energies should be in the form of a polynomial for this application. An interpolation function for a single order parameter should satisfy $h(0) = 1$, $h'(0) = 0$, $h(1) = 1$, and $h'(1) = 0$, for example:

$$h(\eta) = \eta^3(10 - 15\eta + 6\eta^2) \tag{9}$$

The full bulk free energy (for n structural order parameters) is then:

$$f(\eta, c) = f_\alpha(c)\sum_{i=1}^{n}(1 - h(\eta_i)) + f_\beta(c)\sum_{i=1}^{n} h(\eta_i) \tag{10}$$

Equations (7-9) and their derivatives are included in `parameters.h` as follows:

$$\texttt{faV} = f_\alpha(c)$$

$$\texttt{facV} = f'_\alpha(c)$$

$$\texttt{faccV} = f''_\alpha(c)$$

$$\texttt{fbV} = f_\beta(c)$$

$$\texttt{fbcV} = f'_\beta(c)$$

$$\texttt{fbccV} = f''_\beta(c)$$

$$\texttt{hV} = h(\eta)$$

$$\texttt{hnV} = h'(\eta)$$

These free energy variables are incorporated into the residuals in the weak form of equations 4 and 6.

$$\texttt{rcxV} = \Big((1-h(\eta))f''_\alpha(c) + h(\eta)f''_\beta(c)\Big)\nabla c + h'(\eta)(f'_\beta(c) + f'_\alpha(c))\nabla\eta$$

$$\texttt{rnV} = (f_\beta(c) - f_\alpha(c))h'(\eta)$$

$$\texttt{rnxV} = \nabla\eta$$

2.6 `betaPrimePrecipitate` Parameters

The `betaPrimePrecipitate` application combines several features of other applications together to model evolution of coherent precipitates. In practical terms, this application extends `coupledCahnHilliardAllenCahn` by incorporating the strain energy of the transformation into the evolution of the structural order parameter. Following the Landau theory of phase transformations, a stress-free strain tensor (SFST) is defined for each orientation variant (structural order parameter) to be used in the simulation. The total stress-free strain tensor at a point $\mathbf{r}$, $\epsilon^0_{ij}(\mathbf{r})$, is the sum of each order parameter's SFST times it's interpolation function:

$$\epsilon^0_{ij}(\mathbf{r}) = \sum_{p=1}^{n} \epsilon^0_{ij}(p)h(\eta_i(\mathbf{r})) \tag{11}$$

Incorporation of this into the evolution equations is left to the corresponding model documentation.

- `Mc`, `MnVals`, and `Kn0Tensor` are inherited as is from `coupledCahnHilliardAllenCahn`.
- `Ev`, `nuV`, `muV`, `lambdaV`, and `CijklV` are as in `mechanics`
- `sf0StrainV`
  This the stress-free strain tensor (a 9-element vector or 3×3 array) associated with order parameter 0. One SFST is defined for each structural order parameter.
- `skipElasticitySteps`
  The elasticity problem is only re-solved after this many time steps of phase field evolution. To re-solve at every phase field iteration, set this equal to 1.
- `faV`, `facV`, `faccV`, `fbV`, `fbcV`, and `fbccV` are bulk free energy polynomials (and their derivatives) as in `coupledCahnHilliardAllenCahn`.
- `h0V` and `h0nV` are the interpolation function, $h(\eta)$ and its derivative $h'(\eta)$, as in `coupledCahnHilliardAllenCahn`, for order parameter 0. There is an interpolation function defined for each structural order parameter.

- Residuals
  Residuals are defined as in `coupledCahnHilliardAllenCahn`, extended for multiple structural order parameters. More information about the derivation of these can be found in the accompanying Model documentation.

3 Initial Conditions

Initial conditions for the phase field parameters are generated in the functions `InitialConditionN` or `InitialConditionC` located in the `main.cc` file in the application folder. Two types of initial conditions have been implemented so far: a field of random noise around some mean value, and analytical functions of position. Random noise is the default for at least one variable in all of the phase field codes. Here, as an example, c is initialized to random values in the range 0.019-0.021:

```
template <int dim>
double InitialConditionC<dim>::value (const Point<dim> &p, const unsigned int
    /* component */) const
{
       //set result equal to the concentration initial condition
       return 0.02 + 1.0e-3*(2*(0.5 - (double)(std::rand() % 100 )/100.0));
}
```

Analytical functions of position can be generated from the members of the class `p`, e.g. `p.distance` or the x and y coordinates, `p[0]` and `p[1]`. In the default cases, the hyperbolic tangent of the signed distance from a circle or sphere is used. In the following example, η will be initialized into a sphere with radius=spanX/8, where $\eta = 0.5$ at the boundary, $\eta \sim 1$ in the interior, and $\eta \sim 0$ outside.

```
template <int dim>
double InitialConditionN<dim>::value (const Point<dim> &p, const unsigned int
    /* component */) const
{
       //set result equal to the structural order paramter initial condition
       double dx=spanX/std::pow(2.0,refineFactor);
       double r=0.0;
       r=p.distance(Point<dim>(spanX/2.0,spanY/2.0,spanZ/2.0));
       return 0.5*(1.0-std::tanh((r-spanX/8.0)/(3*dx)));
}
```

This example requires that `problemDIM`=3. In the actual applications (e.g. `coupledCahnHilliardAllenCahn`), a conditional tests `problemDIM`, and there are analogous cases for each dimension.

4 Boundary Conditions

Boundary conditions are no-flux for the phase field evolution equations. Implementation of alternate boundary conditions is an objective for future releases.